A FRAMEWORK FOR THE AUTOMATION OF GENERALISED STABILITY THEORY

P. E. FARRELL[*], C. J. COTTER[†], AND S. W. FUNKE[‡]

Abstract. The traditional approach to investigating the stability of a physical system is to linearise the equations about a steady base flow, and to examine the eigenvalues of the linearised operator. Over the past several decades, it has been recognised that this approach only determines the asymptotic stability of the system, and neglects the possibility of transient perturbation growth arising due to the nonnormality of the system. This observation motivated the development of a more powerful generalised stability theory (GST), which focusses instead on the singular value decomposition of the linearised propagator of the system. While GST has had significant successes in understanding the stability of phenomena in geophysical fluid dynamics, its more widespread applicability has been hampered by the fact that computing the SVD requires both the tangent linear operator and its adjoint: deriving the tangent linear and adjoint models is usually a considerable challenge, and manually embedding them inside an eigensolver is laborious. In this paper, we present a framework for the automation of generalised stability theory, which overcomes these difficulties. Given a compact high-level symbolic representation of a finite element discretisation of the nonlinear equations, efficient C++ code is automatically generated to assemble the forward, tangent linear and adjoint models; these models are then used to calculate the optimally growing perturbations to the forward model, and their growth rates. By automating the stability computations, we hope to make these powerful tools a routine part of computational analysis. The efficiency and generality of the framework is demonstrated with applications drawn from geophysical fluid dynamics, phase separation, chemical reaction-diffusion, and quantum mechanics.

Key words. generalised stability theory; adjoint models; tangent linear models; algorithmic differentiation; code generation; finite elements; FEniCS.

AMS subject classifications. 34D10, 34D15, 34D20, 35B20, 25B25, 35B30, 35B35, 74S05

1. Introduction.

The stability of a physical system is a classical problem of mechanics, with contributions from authors such as Lagrange, Dirichlet and Lyapunov [33]. Stability investigates the response of the system to small perturbations applied to a particular initial condition: if for every ϵ there exists a δ-neighbourhood of initial conditions such that their solutions remain within the ϵ-neighbourhood, then the system is stable at that initial condition; otherwise, the system is unstable.

The traditional approach for investigating the stability of physical systems was given by Lyapunov [37]. The nonlinear equations of motion are linearised about a base solution, and the eigenvalues of the linearised system are computed. If all eigenvalues have negative real part, then there exists a finite region of stability around the initial condition: perturbations within that region decay to zero, and the system is asymptotically stable [48].

While this approach has had many successes, several authors have noted that it does not give a complete description of the finite-time stability of a physical system. While the eigendecomposition determines the asymptotic stability of the linearised equations as $t \to \infty$, some systems permit transient perturbations which grow in magnitude, before being predicted to decay. However, if the perturbations grow too large, the linearised equations may cease to hold, and the system may become unstable due to nonlinear effects. More specifically, this transient growth occurs when the system is nonnormal, i.e. when the eigenfunctions of the system do not form an orthogonal basis [52]. For example, Trefethen et al. [57] describe how the traditional approach

[*]Applied Modelling and Computation Group, Department of Earth Science and Engineering, Imperial College London, London, UK, and Center for Biomedical Computing, Simula Research Laboratory, Oslo, Norway (`patrick.farrell@imperial.ac.uk`).

[†]Department of Aeronautics, Imperial College London, London, UK (`colin.cotter@imperial.ac.uk`)

[‡]Applied Modelling and Computation Group, Department of Earth Science and Engineering, and Grantham Institute for Climate Change, Imperial College London, London, UK (`s.funke09@imperial.ac.uk`)

fails to give accurate stability predictions for several classical problems in fluid mechanics, and resolve the problem by analysing the nonnormality of the system in terms of pseudospectra [56].

Therefore, this motivates the development of a finite-time theory of stability, to investigate and predict the transient growth of perturbations. While Lorenz [36] discussed the core ideas (without using modern nomenclature), the development of this so-called generalised stability theory (GST) has been driven by the work of B. F. Farrell and co-workers (e.g., [14, 15, 16, 17]). The main idea is to consider the linearised *propagator* of the system, which is the operator (linearised about the time-dependent trajectory) that maps perturbations in the initial conditions to perturbations in the final state. Essentially, the propagator is the inverse of the tangent linear system associated with the nonlinear forward model, along with operators to load the initial perturbation and select the final perturbation. The perturbations that grow maximally over the time window are given by the singular functions of the propagator associated with the largest singular values. Since the linearised propagator depends on the base solution, it follows that the predictability of the system depends on the conditions of the base solution itself: some states are inherently more predictable than others [36, 29]. This idea has made a significant impact in the meteorological and oceanographic communities, and has been used to investigate many aspects of geophysical fluid dynamics [36, 14, 15, 47, 43, 64, 65]. In the fluid dynamics community, this technique is occasionally referred to as direct optimal growth analysis [5].

While there are some applications of GST in other fields (e.g., [41, 38]), a large number of the applications of this powerful idea have been in the area of geophysical fluid dynamics. One reason for this is that the technique was invented in the meteorological community. Another reason is that nonnormality is important in such flows, whereas traditional eigenvalue analysis is sufficient for the normal case. A final reason is that the necessary adjoint and tangent linear models are commonly available in geophysical fluid dynamics, as they are a necessary component for variational data assimilation, whereas the difficulty of implementing them inhibits the rapid application of GST in other scientific areas. Naumann [44] describes the automatic derivation of efficient adjoint and tangent linear models as "one of the great open challenges of High-Performance Scientific Computing".

The main contribution of this work is a system for automating the calculations required to perform a generalised stability analysis. Given a high-level description of a finite element discretisation of the original time-dependent nonlinear model in the FEniCS framework [34], a representation of the tangent linear and adjoint models in the same high-level format are automatically derived at runtime [19]. These representations are then passed to a finite element form compiler [30], which emits efficient C++ code for the assembly of the nonlinear forward model, the tangent linear model, and its adjoint [35]. The tangent linear and adjoint models are then used automatically in a robust implementation of the Krylov-Schur algorithm [26] for computing a partial singular value decomposition of the model propagator. By automating the difficult steps of deriving the tangent linear model and its adjoint, GST becomes much more accessible: the analyst need only compactly describe a finite element discretisation of the problem of interest, and then can simply request the fastest-growing perturbations and growth rates. The framework presented here is freely available under an open-source license as part of the dolfin-adjoint package (`http://dolfin-adjoint.org`).

This paper is organised as follows. Section 2 gives a brief overview of generalised stability theory, and mentions some applications in the literature. Section 3 describes the main contribution of this paper: how the calculations involved in GST can be entirely automated. This relies on the automatic derivation of tangent linear and adjoint models, as described in section 3.1. Finally, several examples are presented in section 4. The examples are drawn from several areas of computational science to emphasise the widespread applicability of the framework.

2. Generalised stability theory.

2.1. The SVD of the propagator. This presentation of generalised stability theory will consider the stability of the system to perturbations in the initial conditions, but the same approach can be applied to analysing the stability of the system to perturbations in other parameters.

Consider the solution of the model at the final time u_T as a pure function of the initial condition u_0:

$$u_T = M(u_0), \tag{2.1}$$

where M is the *nonlinear propagator* that advances the solution in time over a given finite time window $[0, T]$. Other parameters necessary for the solution (e.g. boundary conditions, material parameters, etc.) are considered fixed. Assuming the model is sufficiently differentiable, the response of the model M to a perturbation δu_0 in u_0 is given by

$$\delta u_T = M(u_0 + \delta u_0) - M(u_0) = \frac{\mathrm{d}M}{\mathrm{d}u_0}\delta u_0 + O\left(||\delta u_0||^2\right). \tag{2.2}$$

Neglecting higher-order terms, the linearised perturbation to the final state is given by

$$\delta u_T \approx \frac{\mathrm{d}M}{\mathrm{d}u_0}\delta u_0 \equiv L\delta u_0, \tag{2.3}$$

where L is the *linearised propagator* (or just propagator) $\mathrm{d}M/\mathrm{d}u_0$ that advances perturbations in the initial conditions to perturbations to the final solution. In the dynamical systems context, L is referred to as the monodromy matrix.

To quantify the stability of the system, we wish to identify perturbations δu_0 that grow the most over the time window $[0, T]$. For simplicity, equip both the initial condition and final solutions with the conventional inner product $\langle \cdot, \cdot \rangle$. We seek the initial perturbation δu_0 of unit norm $||\delta u_0|| = \sqrt{\langle \delta u_0, \delta u_0 \rangle} = 1$ such that

$$\delta u_0 = \underset{||\delta u_0||=1}{\arg\max} \langle \delta u_T, \delta u_T \rangle. \tag{2.4}$$

Expanding δu_T in terms of the propagator,

$$\langle \delta u_T, \delta u_T \rangle = \langle L\delta u_0, L\delta u_0 \rangle = \langle \delta u_0, L^* L\delta u_0 \rangle, \tag{2.5}$$

we see that the leading perturbation is the eigenfunction of L^*L associated with the largest eigenvalue μ, and the growth of the norm of the perturbation is given by $\sqrt{\mu}$. In other words, the leading initial perturbation δu_0 is the leading right singular function of L, the resulting final perturbation δu_T is the associated left singular function, and the growth rate of the perturbation is given by the associated singular value σ. The remaining singular functions offer a similar physical interpretation: if a singular function v has an associated singular value $\sigma > 1$, the perturbation will grow over the finite time window $[0, T]$; if $\sigma < 1$, the perturbation will decay over that time window.

If the initial condition and final solution spaces are equipped with inner products $\langle \cdot, \cdot \rangle_I \equiv \langle \cdot, X_I \cdot \rangle$ and $\langle \cdot, \cdot \rangle_F \equiv \langle \cdot, X_F \cdot \rangle$ respectively, then the leading perturbations are given by the eigenfunctions

$$X_I^{-1} L^* X_F L\delta u_0 = \mu\delta u_0. \tag{2.6}$$

The operators X_I and X_F must be symmetric positive-definite in order to define an inner product. In the finite element context, X_I and X_F are often the mass matrices associated with the input and output spaces, as these matrices induce the functional L_2 norm. All subsequent uses of the term SVD in this paper are taken to include this generalised SVD (2.6).

2.2. Computing the propagator. In general, the nonlinear propagator M that maps initial conditions to final solutions is not available as an explicit function; instead, a PDE is solved. For clarity, let m denote the data supplied for the initial condition. The PDE may be written in the abstract implicit form

$$F(u, m) = 0, \tag{2.7}$$

with the understanding that $u_0 = m$. We assume that for any initial condition m, the PDE (2.7) can be solved for the solution trajectory u; the nonlinear propagator M can then be computed by returning the solution at the final time. Differentiating (2.7) with respect to the initial condition data m yields

$$\frac{\partial F}{\partial u}\frac{\mathrm{d}u}{\mathrm{d}m} = -\frac{\partial F}{\partial m}, \tag{2.8}$$

the *tangent linear system* associated with the PDE (2.7). The term $\partial F/\partial u$ is the PDE operator linearised about the solution trajectory u: therefore, it is linear, even when the original PDE is nonlinear. $\partial F/\partial m$ describes how the equations change as the initial condition data m changes, and acts as the source term for the tangent linear system. $\mathrm{d}u/\mathrm{d}m$ is the prognostic variable of the tangent linear system (2.8), and describes how the solution changes with changes to m. To evaluate the action of the propagator L on a given perturbation δm, the tangent linear system is solved with that particular perturbation, and evaluated at the final time:

$$L\delta m \equiv -\left(\frac{\partial F}{\partial u}\right)^{-1}\frac{\partial F}{\partial m}\delta m\bigg|_T . \tag{2.9}$$

Therefore, to automate the generalised stability analysis of a PDE (2.7), it is necessary to automatically derive and solve the associated tangent linear system (2.8). Furthermore, as discussed in section 3.2, all algorithms for computing the SVD of a matrix A require its adjoint A^*; therefore, it is also necessary to automatically derive and solve the adjoint of the tangent linear system. If the PDE is linear and steady, then this derivation is straightforward; however, if the PDE is nonlinear and time-dependent, the derivation of the associated tangent linear and adjoint systems is widely regarded as a major challenge, even with the assistance of algorithmic differentiation tools [44]. Another crucial concern is the efficiency of the derived models: the SVD computation requires many runs of the tangent linear and adjoint systems, and so their computational performance is of great importance if the stability analysis is to be tractable. However, by exploiting the special structure of finite element discretisations, it is possible to entirely automate the derivation of efficient tangent linear and adjoint models; this is the subject of the next section.

3. Automating generalised stability theory. The following sections explain in detail how the SVD computation is automated by linking the FEniCS framework [34], dolfin-adjoint and SLEPc [26, 25]. An illustration of the role of each software component is given in figure 3.2.

3.1. Automating the generation of tangent linear and adjoint models. This section summarises the novel approach taken for deriving the tangent linear and adjoint models associated with a given PDE solver. The main advantages over traditional approaches are its complete automation, its high performance, and its trivial parallelisation. The approach is more fully described in [19].

The traditional approach to automatically deriving the tangent linear and adjoint models associated with a given PDE solver is to use algorithmic differentiation (AD, also known as automatic differentiation) tools [23, 44]. They primarily operate at the level of the source code

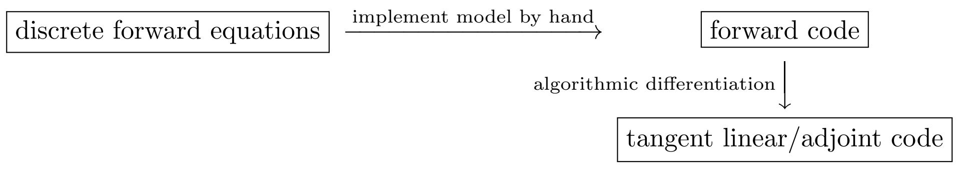
discrete forward equations
implement model by hand
forward code
algorithmic differentiation
tangent linear/adjoint code

Fig. 3.1. *The traditional approach to developing tangent linear and adjoint models. The forward model is implemented by hand, and its adjoint derived either by hand or (more often) with the assistance of an algorithmic differentiation tool.*

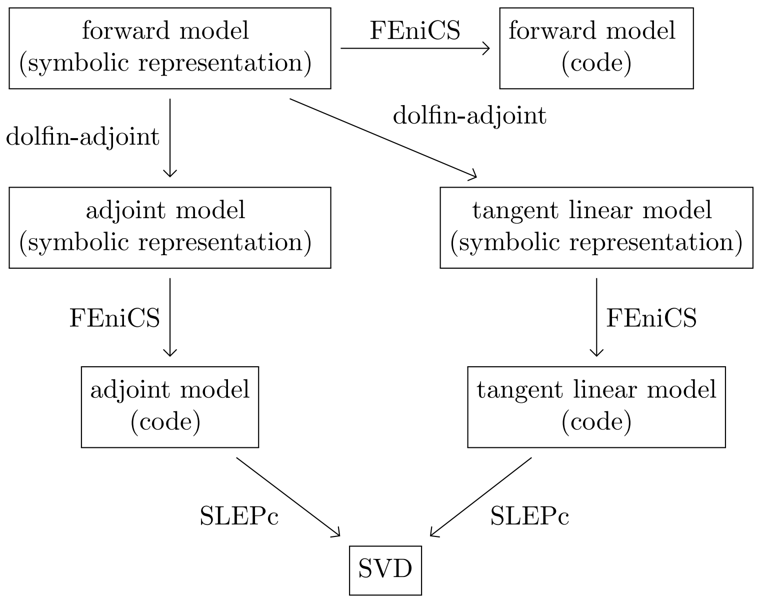
forward model (symbolic representation)
FEniCS
forward model (code)
dolfin-adjoint
dolfin-adjoint
adjoint model (symbolic representation)
tangent linear model (symbolic representation)
FEniCS
FEniCS
adjoint model (code)
tangent linear model (code)
SLEPc
SLEPc
SVD

Fig. 3.2. *The software components for computing the SVD. The user specifies the discrete forward equations in a high-level language similar to mathematical notation; the discrete forward equations are explicitly represented in memory in the UFL format. The in-memory representation of the associated tangent linear and adjoint systems is derived by dolfin-adjoint from the in-memory representation of the forward problem. Both the forward and adjoint equations are then passed to the FEniCS system, which automatically generates and executes the code necessary to compute the forward and adjoint solutions. Finally, SLEPc is used to compute the singular value decomposition.*

(e.g., C++ or Fortran) that implements the discretisation, having already developed the source code by hand. The main idea is to treat the model as a (very long) sequence of elementary instructions, such as additions and multiplications, each of which may be differentiated individually: the derived models are then composed using the chain rule applied forwards (in the tangent linear case) or backwards (in the adjoint case). This approach is sketched in figure 3.1.

Naumann [44] states that “except for relatively simple cases, the differentiation of computer programs is not automatic despite the existence of many reasonably mature AD software packages”. This approach treats the model at a very low level of abstraction, and many of the difficulties of AD stem from this fact.

A source-to-source AD tool operating on the low-level code must parse the source to build a representation of the sequence of elementary instructions as data. This process is inherently fragile. The AD tool must handle complications such as preprocessor directives, parallel directives, libraries for which the source code is not immediately available, expressions with side effects, memory allocation, and aliasing. Correctly and efficiently handling these complications in generality is very difficult, which puts a significant burden on both tool developers and users of algorithmic differentiation.

However, with finite elements, it is possible to circumvent the problem of parsing source code. Finite element methods are based on a powerful high-level abstraction: the language of variational forms. This mathematical abstraction naturally *allows for the discrete equations to be represented as data*. In the FEniCS project [34], the discrete variational form is represented in the Unified Form Language (UFL) format, which is very similar to mathematical notation [1, 2]. This representation is then passed to a specialised finite element form compiler [30], which emits optimised C++ code to assemble the desired discrete equations. This approach has many advantages: it relieves the model developer of much of the manual labour (even complex models such as the Navier-Stokes can be written in tens of lines of code), the form compiler can employ specific optimisations that are complex to perform by hand [45], and the generated code can be tailored to the architecture at a very high level [39].

In the context of stability analysis, this approach has one other major advantage: by representing the equations to be solved as high-level data, the automated derivation of related models (such as the tangent linear and adjoint systems) become much more tractable. This high-level abstraction for the finite element model matches naturally with a higher-level abstraction for model differentiation: our approach takes the view that a model is a sequence of equation solves. This approach is implemented in the dolfin-adjoint software package [19]. When the dolfin-adjoint module is imported, all functions that solve equations or modify variable values are overloaded. These overloaded functions build a tape of the forward model at runtime; this tape is analogous to the concept of a tape in algorithmic differentiation, except that the units on the tape represent entire equation solves stored in UFL, rather than elementary operations. This tape may then be used to derive the associated tangent linear and adjoint models, even for complex nonlinear time-dependent systems. By coupling the high-level representation of the forward model with this high-level differentiation approach, the tangent linear and adjoint versions of a model written in the FEniCS framework may be derived with almost no user intervention or effort [19]. With the dolfin-adjoint software package, the discrete tangent linear and adjoint equations to be solved are symbolically derived in the exact same UFL format as the forward model, and passed to the same finite element compiler.

This alternative approach to automating the derivation of the tangent linear and adjoint models has several major advantages for generalised stability analysis. Firstly, the derivation of the tangent linear and adjoint models is almost entirely automatic. In the example shown in section 3.3, the user need only add two lines of code: one to import the dolfin-adjoint library, and one to request the leading singular triplets. Secondly, the derived tangent linear and adjoint models approach optimal theoretical efficiency. This is crucial, as the SVD calculation requires many iterations of the tangent linear and adjoint models; the efficiency of the approach will be demonstrated on several examples in section 4. Thirdly, whereas applying algorithmic differentiation to a parallel code is a major research challenge [61, 20], this high-level approach parallelises very naturally: if the forward model runs in parallel, the tangent linear and adjoint models will also [19]. In fact, there is no parallel-specific code in dolfin-adjoint – by operating on the discrete equations instead of the source code, the problem of parallelisation dissolves. As the computational demands in problems of practical interest are usually very large, parallelisation is a necessity if the GST framework is to be used in such cases.

3.2. Singular value decomposition. Once the propagator L is available, its singular value decomposition may be computed. There are two main computational approaches. The first approach is to compute the eigendecomposition of the *cross product* matrix L^*L (or LL^*, whichever is smaller). The second is to compute the eigendecomposition of the *cyclic* matrix

$$H(L) = \begin{pmatrix} 0 & L \\ L^* & 0 \end{pmatrix}. \tag{3.1}$$

The latter option is more accurate for computing the small singular values, but is more expensive [55]. As we are only interested in a small number of the largest singular triplets, the cross product approach is used throughout this work. Note that regardless of which approach is taken, the adjoint propagator L^* is necessary to compute the SVD of L.

The algorithm used to compute the eigendecomposition of the cross product matrix is the Krylov-Schur algorithm [54], as implemented in SLEPc [26, 25]. As the cross product matrix is Hermitian, this algorithm reduces to the thick-restart variant [63] of the Lanczos method [32]. This algorithm was found experimentally to be faster than all other algorithms implemented in SLEPc for the computation of a small number of singular triplets, which is the case of interest in stability analysis.

Rather than computing and storing a dense matrix representation of the propagator, the action of the propagator is computed in a matrix-free fashion, using the tangent linear model. In turn, the entire time-dependent tangent linear model is not stored, but its action is implemented as the solution of several equations in sequence. In turn, the solution of each equation may optionally be achieved in a matrix-free fashion; the automatic derivation of the tangent linear and adjoint systems supports such an approach [19]. Similarly, the adjoint propagator is computed in a matrix-free fashion using the adjoint model. SLEPc elegantly supports such matrix-free computations through the use of PETSc shell matrices [3, 4].

3.3. Code example. In order to demonstrate the user interface of the proposed framework, a code example for a generalised stability analysis of the nonlinear Burgers' equation is given in figure 3.3. The example is complete; nothing has been removed. Only two lines of code are added to the forward model to conduct the GST: one to import the dolfin-adjoint library, and one to perform the GST computation. The `compute_gst` function symbolically derives the tangent linear and adjoint models, generates the shell matrix for the propagator, and employs that inside a Krylov-Schur algorithm for computing the SVD, with the FEniCS framework automatically generating the code to assemble the discrete systems as necessary. By exploiting the high-level symbolic representation of the problem, the code can be very compact and readable. This makes configuring a generalised stability analysis much more accessible to computational scientists and engineers.

4. Verification and applications.

All applications are available under an open-source license as part of the dolfin-adjoint applications repository (`http://dolfin-adjoint.org`). In all of the examples, the mass matrices of the input and output spaces were used to define the norms in equation (2.6). The benchmark tables show the minimum time of five experiments, performed on 8 2.13 GHz Intel Xeon CPU cores with 12 GB memory.

4.1. Verification: the nonlinear Burgers' equation. The verification of the framework proceeds in two stages. Firstly, the correctness of the tangent linear and adjoint models must be verified. Secondly, the correctness of the singular value decomposition must be verified.

The fundamental tool in verifying the correctness of the tangent linear and adjoint models is the Taylor remainder test. Suppose we have a black box for evaluating a function $f(x)$, and have a candidate function for its gradient ∇f. The correctness of the gradient can be asserted by noting that by Taylor's theorem, the first order Taylor remainder

$$|f(x + h\delta x) - f(x)| \to 0 \quad \text{at } O(h) \tag{4.1}$$

```
 1 from dolfin import *
 2 # Import the dolfin-adjoint library to enable the derivation
 3 # of tangent linear and adjoint models
 4 from dolfin_adjoint import *
 5 
 6 # Define the computational domain and function spaces
 7 mesh = UnitInterval(10)
 8 V = FunctionSpace(mesh, "Lagrange", 1)
 9 
10 # Define the necessary test and trial functions
11 ic = project(Expression("sin(2*pi*x[0])"), V)
12 u = Function(ic, name="State")
13 u_next = Function(V, name="NextState")
14 v = TestFunction(V)
15 
16 # Define the viscosity and timestep
17 nu = Constant(0.0001)
18 timestep = Constant(0.1)
19 
20 # Define the weak formulation of the Burger's equation
21 F = (((u_next - u)/timestep)*v
22      + u_next*grad(u_next)*v + nu*grad(u_next)*grad(v))*dx
23 bc = DirichletBC(V, 0.0, "on_boundary")
24 
25 # Run the forward model
26 t = 0.0
27 end = 0.2
28 while (t <= end):
29     solve(F == 0, u_next, bc)
30     u.assign(u_next)
31 
32     t += float(timestep)
33 
34 # Compute the five largest singular values for the propagator
35 # that maps the initial state of the Burger's solution
36 gst = compute_gst(ic="State", final="State", nsv=5)
```

FIG. 3.3. *The entire code to compute a generalised stability analysis of the nonlinear Burgers' equation. This code uses piecewise linear Lagrange finite elements for the spatial discretisation (lines 8, 21–22), and implicit Euler for the temporal discretisation (lines 21–22). The high-level approach leads to extremely compact and readable code. In order to use the framework presented here, only two additional lines are necessary (in blue): one to import the dolfin-adjoint library (line 4), and one to compute the singular value decomposition of the propagator associated with the forward model (line 36). The functions in red are overloaded by dolfin-adjoint in order to record the information necessary for the derivation of the tangent linear and adjoint models as described in section 3.1. The* `compute_gst` *function (line 36) symbolically derives the tangent linear and adjoint models, creates a shell matrix to compute the action of the propagator, and embeds it inside a Krylov-Schur algorithm to compute the requested number of singular triplets.*

converges to zero at first order, but that the Taylor remainder corrected with the gradient

$$\left|f(x + h\delta x) - f(x) - h\delta x^T \nabla f\right| \to 0 \quad \text{at } O(h^2) \tag{4.2}$$

converges to zero at second order. In this context, the function $f(u)$ is a functional of the solution u of a PDE system $F(u, m) = 0$ specified by parameters m, and its gradient $\nabla_m f(u(m))$ is computed in two different ways, once using the tangent linear model and once using its adjoint.

For the verification exercise, we choose as our model the nonlinear time-dependent Burgers' equation:

$$F(u, m) \equiv \frac{\partial u}{\partial t} + u \cdot \nabla u - \nu \nabla^2 u = 0, \tag{4.3}$$

on some domain $\Omega \times [0, T]$, along with suitable boundary conditions and diffusivity coefficient ν. The parameter m is the initial condition for u. We choose our functional J as

$$J(u) = \int_\Omega \langle u_T, u_T \rangle \ \mathrm{d}x, \tag{4.4}$$

the square of the L_2 norm of the solution evaluated at the end of time. By the chain rule, the gradient $\mathrm{d}J(u(m))/\mathrm{d}m$ can be computed with

$$\frac{\mathrm{d}J(u(m))}{\mathrm{d}m} = \left\langle \frac{\partial J}{\partial u}, \frac{\mathrm{d}u}{\mathrm{d}m} \right\rangle, \tag{4.5}$$

where $\mathrm{d}u/\mathrm{d}m$ is the solution of the associated tangent linear system (2.8). In this way, the automated derivation of the tangent linear system (2.8) from the nonlinear forward model (4.3) can be rigorously verified: the tangent linear solution is correct if and only if the second order Taylor remainder (4.2) converges at second order. In practice, computing the whole of the solution Jacobian $\mathrm{d}u/\mathrm{d}m$ is unnecessary, as we only require the *action* of the gradient $\mathrm{d}J/\mathrm{d}m$ on a particular perturbation $h\delta m$. In this case, it is sufficient to compute

$$\left\langle \frac{\mathrm{d}J(u(m))}{\mathrm{d}m}, h\delta m \right\rangle = \left\langle \frac{\partial J}{\partial u}, h\frac{\mathrm{d}u}{\mathrm{d}m}\delta m \right\rangle, \tag{4.6}$$

where the action of the solution Jacobian $\mathrm{d}u/\mathrm{d}m$ on the perturbation $h\delta m$ is computed via

$$\frac{\partial F}{\partial u}\left(h\frac{\mathrm{d}u}{\mathrm{d}m}\delta m\right) = -h\frac{\partial F}{\partial m}\delta m. \tag{4.7}$$

The Burgers' equation (4.3) is discretised in space using standard piecewise quadratic finite elements and discretised in time using Crank-Nicolson timestepping [11], and the resulting nonlinear system solved via Newton iteration. As described in section 3, the tangent linear model is automatically derived, with almost no user intervention. The results of the Taylor remainder test for the tangent linear model can be seen in table 4.1. As expected, the Taylor remainders corrected with the functional gradient do indeed converge at second order, indicating that the computed gradient, the tangent linear solution, and the tangent linear equations are all correct.

Similarly, the adjoint model may be verified, by computing the gradient $\mathrm{d}J/\mathrm{d}m$ via the relation

$$\frac{\mathrm{d}J(u(m))}{\mathrm{d}m} = -\left\langle \lambda, \frac{\partial F}{\partial m} \right\rangle, \tag{4.8}$$

h	$\left\|\widehat{J}(\tilde{m}) - \widehat{J}(m_0)\right\|$	order	$\left\|\widehat{J}(\tilde{m}) - \widehat{J}(m_0) - \tilde{m}^T\nabla\widehat{J}\right\|$	order
1×10^{-3}	1.8664×10^{-5}		5.8991×10^{-7}	
5×10^{-4}	9.4796×10^{-6}	0.9773	1.4747×10^{7}	2.000
2.5×10^{-4}	4.7766×10^{-6}	0.9888	3.6868×10^{-8}	2.000
1.25×10^{-4}	2.3975×10^{-6}	0.9944	9.2169×10^{-9}	2.000
6.25×10^{-5}	1.2010×10^{-6}	0.9972	2.3042×10^{-9}	2.000

TABLE 4.1

Verification of the tangent linear model. The Taylor remainders for the functional $\widehat{J} = J(u(m))$ are evaluated at a perturbed initial condition $\tilde{m} \equiv m_0 + h\delta m$, where the perturbation direction δm is pseudorandomly generated. As expected, the Taylor remainder incorporating gradient information computed using the tangent linear model converges at second order, indicating that the functional gradient computed using the tangent linear model is correct.

h	$\left\|\widehat{J}(\tilde{m}) - \widehat{J}(m_0)\right\|$	order	$\left\|\widehat{J}(\tilde{m}) - \widehat{J}(m_0) - \tilde{m}^T\nabla\widehat{J}\right\|$	order
1×10^{-3}	4.0880×10^{-5}		9.5164×10^{-7}	
5×10^{-4}	2.0678×10^{-5}	0.9833	2.3786×10^{7}	2.000
2.5×10^{-4}	1.0398×10^{-5}	0.9917	5.9459×10^{-8}	2.000
1.25×10^{-4}	5.2141×10^{-6}	0.9958	1.4864×10^{-9}	2.000
6.25×10^{-5}	2.6107×10^{-6}	0.9979	3.7159×10^{-9}	2.000

TABLE 4.2

Verification of the adjoint model. The Taylor remainders for the functional $\widehat{J} = J(u(m))$ are evaluated at a perturbed initial condition $\tilde{m} \equiv m_0 + h\delta m$, where the perturbation direction δm is pseudorandomly generated. As expected, the Taylor remainder incorporating gradient information computed using the adjoint model converges at second order, indicating that the functional gradient computed using the adjoint model is correct.

where λ is the solution of the adjoint equation

$$\left(\frac{\partial F}{\partial u}\right)^* \lambda = \frac{\partial J}{\partial u}. \tag{4.9}$$

The results of the Taylor remainder test for the adjoint model can be seen in table 4.2. Again, the Taylor remainders corrected with the functional gradient do indeed converge at second order, indicating that the computed gradient, the adjoint solution, and the automatically derived adjoint equations are all correct.

With the correctness of the tangent linear and adjoint models established, the correctness of the singular value decomposition was verified. As described in section 3.2, in practical computations the propagator is never represented as a matrix: instead, its action is computed using the tangent linear model. However, for verification purposes, the full singular value decomposition of the propagator was computed, and a dense matrix representation of the propagator was generated by multiplying the output matrices together. This calculation was expensive, and unnecessary in the general case: the computation was performed merely for the purposes of verification. The action of the propagator was computed in two independent ways (matrix-free with the tangent linear model, and with the dense representation), and the results were found to be identical to within machine precision, confirming the accuracy of the singular value decomposition.

Finally, the relevance of the computed SVD was verified by running the nonlinear forward model with the initial condition perturbed with the leading right singular vector. The actual growth rate of the perturbation was compared with the growth rate predicted from the singular

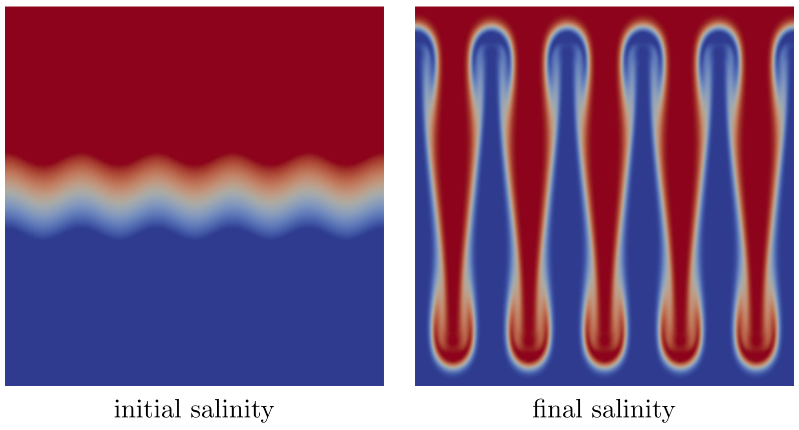

FIG. 4.1. *The phenomenon of salt fingering. Warm salty water overlies cold fresh water. If a parcel of warm salty water sinks downwards into the colder region, the heat of the parcel is diffused away much faster than its salt, thus making the parcel denser, and causing it to sink further. Left: the initial condition for salinity, using the perturbed interface of [46]. Right: the final salinity, after fifty timesteps.*

value; the prediction matched the actual growth rate to within 1%. This confirms the physical utility of the SVD for predicting the dynamics of small perturbations to the initial condition.

4.2. Navier-Stokes: double-diffusive salt fingering. In the ocean, the diffusivity coefficient of temperature is approximately two orders of magnitude larger than the diffusivity coefficient of salinity. Suppose warm salty water lies above colder, less salty water. If a parcel of warm salty water sinks downwards into the colder region, the heat of the parcel will diffuse away much faster than its salt, thus making the parcel denser, and causing it to sink further. Similarly, if a parcel of cold, less salty water rises into the warmer region, it will gain heat from its surroundings much faster than it will gain salinity, making the parcel more buoyant. This phenomenon is referred to as "salt fingering" [53] (figure 4.1) and has been observed in many real-world oceanographic contexts [59]. An initial investigation of this phenomenon using the tools of generalised stability theory was presented in [13].

Özgökmen [46] used a numerical model to investigate asymmetry in the growth of salt fingers caused by nonlinearities in the equation of state. In this work, we investigate the stability of the proposed configuration to small perturbations, and examine what this means for its utility as a numerical benchmark. The two-dimensional vorticity-streamfunction formulation of the Navier-Stokes equations is coupled to two advection equations for temperature and salinity:

$$\frac{\partial \zeta}{\partial t} + \nabla^{\perp}\psi \cdot \nabla \zeta = \frac{\mathrm{Ra}}{\mathrm{Pr}}\left(\frac{\partial T}{\partial x} - \frac{1}{R_\rho^0}\frac{\partial S}{\partial x}\right) + \nabla^2 \zeta, \tag{4.10}$$

$$\frac{\partial T}{\partial t} + \nabla^{\perp}\psi \cdot \nabla T = \frac{1}{\mathrm{Pr}}\nabla^2 T, \tag{4.11}$$

$$\frac{\partial S}{\partial t} + \nabla^{\perp}\psi \cdot \nabla S = \frac{1}{\mathrm{Sc}}\nabla^2 S, \tag{4.12}$$

$$\nabla^2 \psi = \zeta, \tag{4.13}$$

where ζ is the vorticity, ψ is the streamfunction, T is the temperature, S is the salinity, and Ra, Sc, Pr and R_ρ^0 are nondimensional parameters. Periodic boundary conditions are applied on the left and right boundaries; for full details of the remaining boundary conditions and values of the numerical parameters, see [46]. The configuration consists of two well-mixed layers (i.e.,

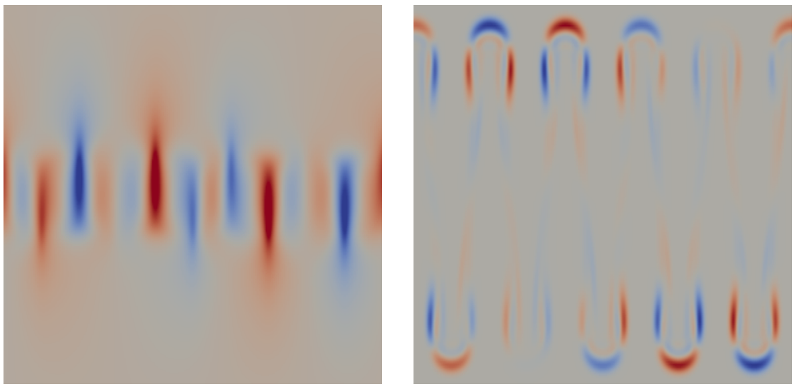

Fig. 4.2. *The leading perturbation to the salt fingering system. When the perturbation on the left is applied to the initial condition for salinity, the perturbation grows with a growth factor $\sigma \approx 1553$, resulting in a much larger perturbation to the final salinity.*

	Runtime (s)	Ratio
Forward model	165.89	
Tangent linear model (averaged)	65.25	1.39
Adjoint model (averaged)	68.71	1.41

Table 4.3

Timings for the salt fingering simulation for computing the perturbation that grows optimally to $T = 0.05$. The optimal perturbation is obtained after 24 tangent linear and adjoint model solves. The table shows the run time for the forward and the averaged timings for the tangent linear and adjoint solves. As can be seen, the tangent linear and the adjoint models take approximately 40% of the cost of the forward model. The optimal ratio is approximately 1.33.

of homogeneous temperature and salinity) separated by an interface. To activate the instability, [46] added a sinusoidal perturbation to the initial salinity field (figure 4.1).

To investigate the possibility of a secondary instability about this perturbed initial condition, the framework of generalised stability theory was applied. The PDE was discretised in space using standard piecewise linear finite elements, and θ-timestepping was employed in time with $\theta = 0.6$. The solution trajectory was computed using the initial sinusoidal perturbation to the salinity field, the propagator was linearised about that trajectory, and the leading ten singular triplets were computed. This calculation was repeated on several refinements of a structured mesh, up to 300×300, to ensure the impact of the numerical discretisation was negligible for the purpose of physical analysis.

The leading input perturbation is plotted in figure 4.2, along with the resulting linear perturbation to the final state. As visible in the figure, the leading perturbation encourages the growth of some fingers, while retarding the growth of others. We identified a number of unstable modes which result in an uneven distribution of salt finger lengths; the physical mechanism is that longer fingers retard the growth of the shorter fingers since incompressibility requires a return flow in the opposite direction either side of each finger. All ten perturbations computed were found to grow over the time interval $[0, 0.05]$; the leading perturbation grew in norm by a factor of approximately 1553 over the time window. To the best of our knowledge, this is the first time that this secondary instability has been documented.

The performance was benchmarked by recording the run times of the forward, tangent linear

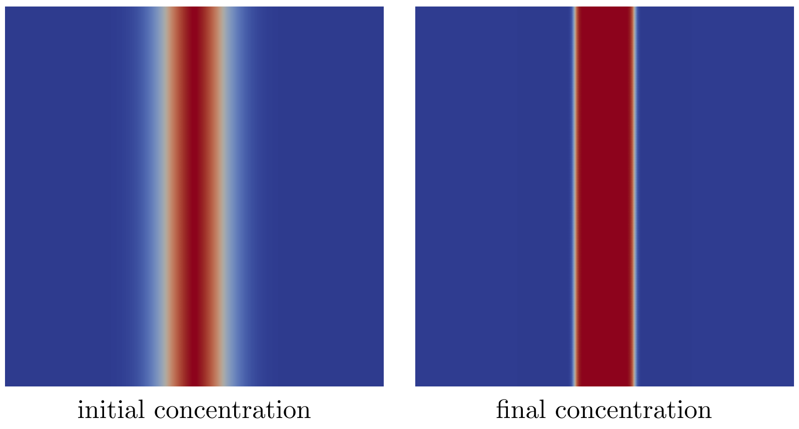

Fig. 4.3. *The initial and final conditions for the Cahn-Hilliard simulation. The color bar ranges from 0 to 1.*

and adjoint models. The numerical results can be seen in table 4.3. During the forward solve, the Newton solver typically converges after three iterations. As both the adjoint and the tangent linear model replace each Newton solve with one linear solve, a coarse estimate of the optimal performance is that the tangent linear and adjoint models should take 33% of the run time of the forward model, for an optimal ratio of 1.33. (Efficiency results for derived models always include the cost of the forward model also, as running the forward model is necessary to run derived models [44]). The numerical results yield a value of approximately 40% of the cost of the forward model; the tangent linear and the adjoint models approach optimal performance.

4.3. Cahn-Hilliard: phase separation. The Cahn-Hilliard equation is a partial differential equation which describes the process of phase separation, in which two components of a mixed binary fluid separate to form pure regions of each component [8]. The equation has also found applications in image processing, for evolving object contours [9], and astrophysics, for modelling the evolution of Saturn's rings [58]. The Cahn-Hilliard equation is a nonlinear fourth-order parabolic equation:

$$\frac{\partial c}{\partial t} - \nabla \cdot M \left(\nabla \left(\frac{\mathrm{d}f}{\mathrm{d}c} - \lambda \nabla^2 c \right) \right) = 0 \quad \text{on } \Omega, \tag{4.14}$$

$$M \left(\nabla \left(\frac{\mathrm{d}f}{\mathrm{d}c} - \lambda \nabla^2 c \right) \right) = 0 \quad \text{on } \partial\Omega, \tag{4.15}$$

$$M \lambda \nabla c \cdot n = 0 \quad \text{on } \partial\Omega, \tag{4.16}$$

where c is the prognostic concentration field ($c = 1$ is one fluid, $c = 0$ the other), f is the (prescribed) chemical potential, n is the outward unit normal, and λ and M are scalar constants. In order to apply standard continuous finite elements, the fourth-order equation is broken up into two coupled second-order equations, and a mixed P1-P1 finite element discretisation applied [62].

Generalised stability analysis was employed to investigate the stability of the evolution of the Cahn-Hilliard system from a randomly perturbed initial condition on the domain $\Omega = [0, 2]^2$. The initial condition was given by the one-dimensional profile

$$c_0 = c(t = 0) = e^{-30(x-1)^2}. \tag{4.17}$$

The constants were set to $\lambda = 10^{-2}$ and $M = 1$, and $f = 100c^2(1-c)^2$. The initial (at $t = 0$) and final conditions (at $t = 5 \times 10^{-4}$) for the simulation are presented in figure 4.3. The mesh had

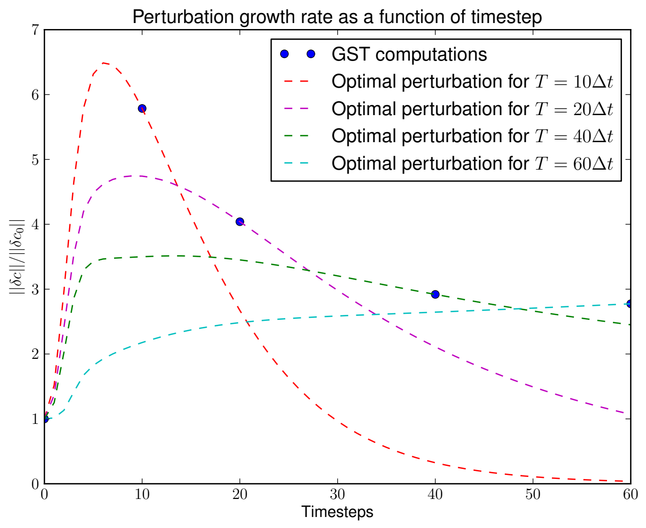

Fig. 4.4. *The growth rate of the optimal perturbation computed using GST at various times (blue dots), and the growth rate of the optimal perturbation associated with various timesteps, computed using the nonlinear model (dashed lines). To compute the dashed curves, the identified perturbation was scaled to have norm $||\delta c_0|| = 10^{-7}$, and was added to unperturbed initial condition. The nonlinear model was then executed with this perturbed initial condition, and the results compared to the original unperturbed nonlinear trajectory. The fact that the dashed curves (observed from the nonlinear model) match the GST predictions indicates that the GST analysis is correct.*

	Runtime (s)	Ratio
Forward model	66.63	
Tangent linear model (averaged)	17.64	1.26
Adjoint model (averaged)	17.92	1.27

Table 4.4

Timings for the Cahn-Hilliard simulation for computing the perturbation that grows optimally to $T = 10\Delta t$. The perturbation is obtained after 72 tangent linear and adjoint model solves. The table shows the run time for the forward and the averaged timings for the tangent linear and adjoint solves. The optimal ratio is approximately 1.25.

150 elements in both the $x-$ and $y-$ directions, leading to a mixed function space with 90602 degrees of freedom. The timestep Δt was set to 5×10^{-6}. The simulations were run in parallel across 8 cores using MPI.

The generalised stability analysis was used to compute the optimally linearly growing perturbations to the initial condition for concentration and their growth rates at times $T = 10\Delta t$, $20\Delta t$, $40\Delta t$, and $60\Delta t$. The optimal growth rate computed using GST as a function of timestep is shown in figure 4.4 (solid blue dots). In general, the perturbation that grows optimally to a time T_1 will be different to the perturbation that grows optimally to a time $T_2 \neq T_1$; that is, the singular vectors are sensitive to the integration period of the propagator [29, pg. 220]. This is indeed the case for the GST analysis of the Cahn-Hilliard system. The leading singular vectors of the propagator defined with respect to various times is shown in figure 4.5.

To further verify the utility of GST, the nonlinear model was perturbed with each identified optimal perturbation, in order to compare the growth rates predicted by the GST with

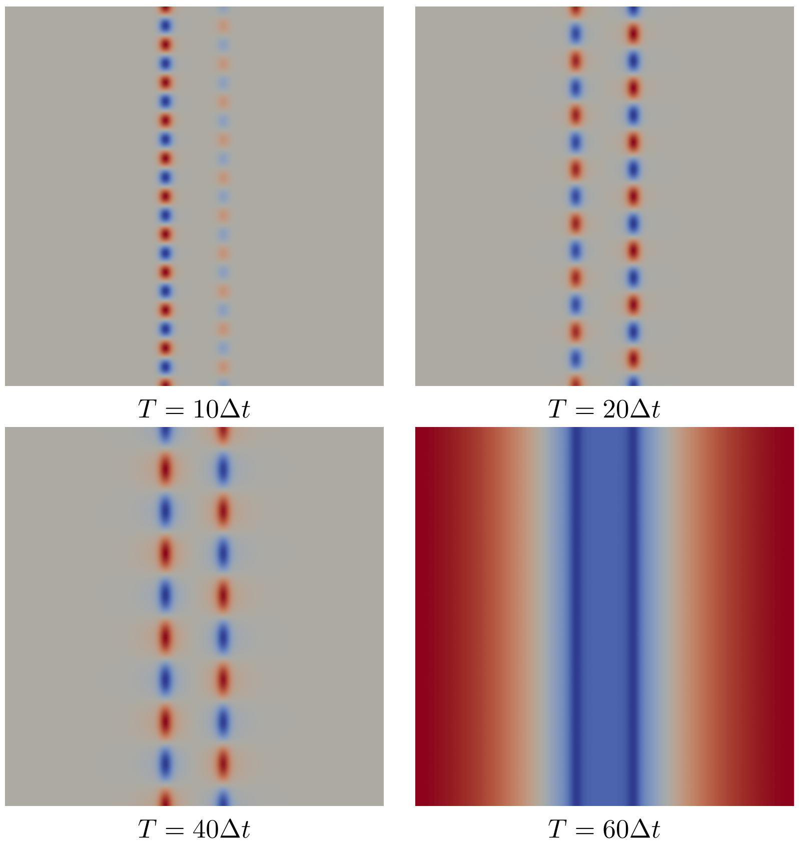

FIG. 4.5. *The perturbation to the Cahn-Hilliard concentration that grows optimally (equivalently, the leading singular vector of the propagator), displayed for various integration periods. As the propagator is linear by definition, the scales of the perturbations do not matter, and so the perturbations are normalised to have unit norm. The optimal perturbation depends sensitively on the time to which the propagator is defined.*

the actual growth rates observed. The predictions and observations match closely, indicating that the GST is indeed predicting the quantitative behaviour of the system (figure 4.4, dashed lines). The growth curves of the perturbations demonstrate the phenomenon of transient growth: initial growth in magnitude over some finite time horizon, followed by asymptotic decay. Such phenomena are characteristic of nonnormal systems [56].

The run times of the forward, tangent linear and adjoint models for the setup with $T = 10\Delta t$ are shown in table 4.4. For this configuration, the Newton solver typically converges after four iterations during the forward simulation. Therefore, the optimal performance can be estimated to be 25% of the run time of the forward model, for an optimal ratio of 1.25. The benchmark results yield a value of 27% of the cost of the forward model; the tangent linear and adjoint models approach optimal efficiency.

4.4. Gray-Scott: reaction-diffusion. The Gray-Scott equations model the reaction and diffusion of two chemical species, U and V [21, 22]. They have attracted considerable mathematical interest due to the variety and complexity of the patterns that arise in their solution, including stripes, spots, rolls and Turing patterns [49].

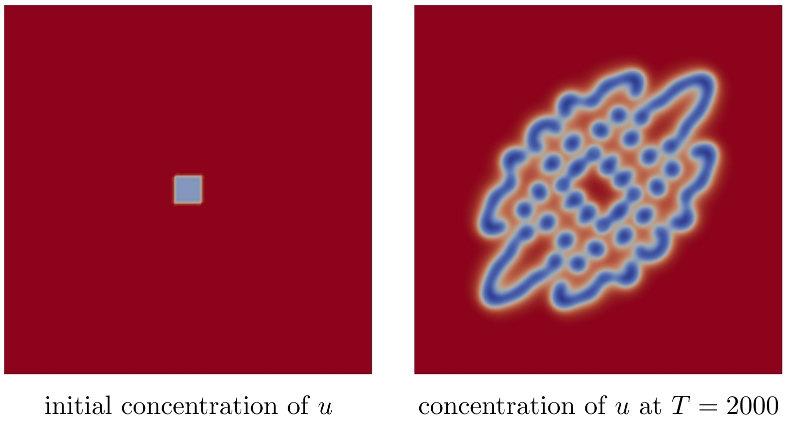

FIG. 4.6. *The forward solution u of the Gray-Scott equations. The initial perturbation propagates slowly over the domain, eventually reaching a quasi steady-state at approximately $T = 20000$ (not shown). The color bar ranges from 1 to 0.34.*

	Runtime (s)	Ratio
Forward model	13.84	
Tangent linear model (averaged)	14.00	2.01
Adjoint model (averaged)	14.42	2.04

TABLE 4.5
Timings for the Gray-Scott simulation for computing the perturbation that grows optimally to $T = 20\Delta t$ and a time step of 0.25. The perturbation is obtained after 24 tangent linear and adjoint model solves. The table shows the run time for the forward and the averaged timings for the tangent linear and adjoint solves. The optimal ratio is approximately 2.

The configuration considered here is adopted from [49]. The Gray-Scott equations are

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u - uv^2 + F(1-u), \tag{4.18}$$

$$\frac{\partial v}{\partial t} = D_v \nabla^2 v + uv^2 + (F+k)v, \tag{4.19}$$

where $u = [U]$ and $v = [V]$ are the prognostic concentrations of the two chemical species, D_u and D_v are diffusion coefficients, F is the feed rate, and k is the removal rate for v.

The equations were solved on a two-dimensional domain $\Omega = [0.0, 2.5]^2$, with periodic boundary conditions imposed. Following [49], the initial conditions were constructed as follows. First u and v were set to 1 and 0, respectively. Then, the values inside the centered square with length 0.1 were perturbed to 1/2 for u and 1/4 for v. Finally, these conditions were perturbed with pseudorandom noise over the whole domain uniformly distributed on $[0, 0.01]$, to break the symmetry of the square. The diffusion coefficients were set to $D_u = 2 \cdot 10^{-5}$ and $D_v = 10^{-5}$. The solution patterns highly depends on the removal rate k and the feed rate F. The experiments presented here were performed in parallel with $k = 0.055$ and $F = 0.02$, for which [49] observed a dotted solution.

The equations were solved using piecewise linear functions and the time was discretised with the explicit Euler method. The initial perturbation slowly spreads over the domain and reaches a quasi-steady state at approximately $T = 20,000$ timesteps (figure 4.6).

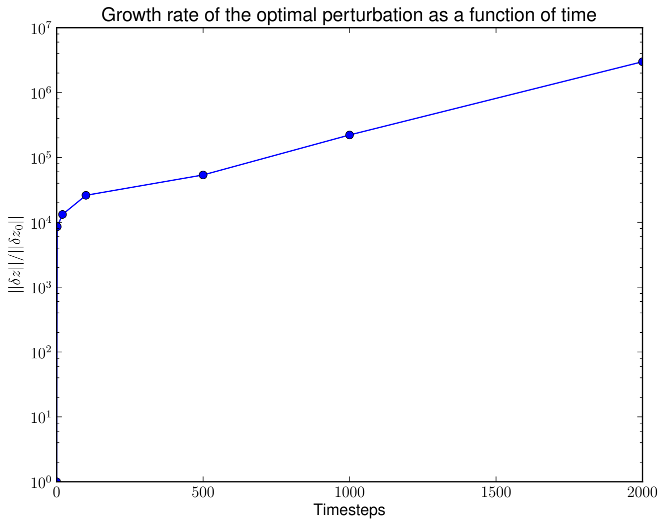

FIG. 4.7. *The growth rate of the optimal perturbation for the Gray-Scott system as a function of timestep. Both axes are dimensionless; the y-axis is plotted with a logarithmic scale. The simulation is unstable to perturbations in the initial conditions.*

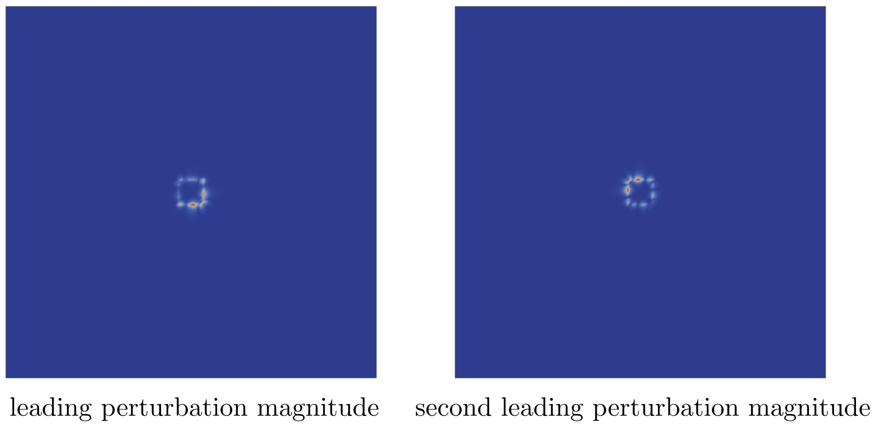

FIG. 4.8. *The perturbations to the Gray-Scott initial condition that grow optimally for $T = 2000$ (equivalently, the first and second singular vectors of the propagator). The perturbations are normalised to have unit norm. The optimal perturbations are localised to the square, to promote preferential propagation in certain directions.*

Generalised stability analysis was applied to investigate the stability of this initial condition to secondary perturbations. Let $z = (u, v)$ with norm $||z|| = ||u|| + ||v||$, and let z_0 denote the initial condition described above. The optimal growth rate as a function of timestep is shown in figure 4.7. Similar to the Cahn-Hilliard case, the optimal secondary perturbation is a function of time, but all optimal secondary perturbations share the common feature that the secondary perturbation is localised to the square, to preferentially promote the propagation of the initial square perturbation in certain directions. An example for $T = 2000$ is shown in figure 4.8.

The timing results are given in table 4.5. Since an explicit time discretisation is employed, no Newton iterations are involved in the forward solution process, and so the optimal ratio is 2.

4.5. Gross-Pitaevskii: soliton solutions. The Gross-Pitaevskii equation [24, 50] is a nonlinear Schrödinger equation that describes the dynamics of a quantum system of identical bosons. The nondimensional equation governing the evolution of the wavefunction Ψ is given by

$$i\frac{\partial \Psi}{\partial t} + \nabla^2 \Psi + s|\Psi|^2\Psi = 0, \tag{4.20}$$

where s is a parameter ($s = 1$ is the focussing case, $s = -1$ the defocussing case). In particular, the Gross-Pitaevskii equation describes the behaviour of Bose-Einstein condensates, a state of matter observed when a dilute gas of bosons is cooled to temperatures close to absolute zero [7, 12]. Bose-Einstein condensates are of considerable interest as they permit black hole analogues: systems from which acoustic perturbations, rather than light, are unable to escape [60]. This could potentially allow the laboratory-scale experimental investigation of the physics of black holes [18, 31].

Generalised stability theory was employed to investigate the stability of the one-dimensional soliton solution of the focussing Gross-Pitaevskii equation

$$\Psi = \sqrt{2}\frac{\exp\left(\frac{i}{2}x + \frac{3i}{4}t\right)}{\cosh\left(x - t\right)} \tag{4.21}$$

to perturbations in the initial condition. The Gross-Pitaevskii equation was solved with piecewise linear finite elements on the domain $\Omega = [-10, 10]$ with periodic boundary conditions applied. The initial condition was achieved by pointwise evaluation of (4.21), and the equations were advanced in time from 0 to T using the implicit midpoint rule. The interval was discretised with $N = 480$ elements, and the timestep was set to $\Delta t = 0.03125$.

The results of the GST calculation for various times are shown in figure 4.9a. In this example, approximately linear growth of the optimal perturbations is observed. For $T > 10$, all GST calculations yielded the same optimal perturbation (shown in figure 4.9b).

This optimal perturbation corresponds to shifting along the family of soliton solutions parameterised by their amplitude. Since each member of this family has a different speed, perturbing in this direction leads to a similar shaped soliton moving at a different speed, hence the linear growth in the perturbation. The fact that this is the fastest growing perturbation indicates that the soliton solutions are stable. This is illustrated in figure 4.10.

The timing results are given in table 4.5. For this example, the model only has one spatial dimension which makes the linear solves computationally cheap. As a consequence, the cost of the linear solves does not dominate the cost of the symbolic manipulation for low resolutions ($N = 480$), and so the efficiency ratio is suboptimal. However, as the mesh resolution is increased ($N = 12,000$), the cost of the linear solves increases while the cost of the symbolic manipulation does not. Therefore as the mesh is refined, the efficiency ratio approaches the optimal value.

5. Conclusions. Generalised stability theory is a powerful tool for investigating the dynamics of physical systems, but the difficulty of implementing it has been a major impediment to its widespread application. The core contribution of this paper has been to remove this barrier. By employing a new high-level symbolic approach to automating the derivation of adjoint and tangent linear models, conducting a generalised stability analysis is now straightforward, even for parallel discretisations of complex nonlinear coupled time-dependent problems. The widespread applicability of the framework was demonstrated on examples drawn from geophysical fluid dynamics, phase separation, chemical reaction-diffusion, and quantum mechanics.

Adjoint and tangent linear models arise across computational mathematics, not merely in stability analysis. Therefore, the same core technology of the automated derivation of adjoint and

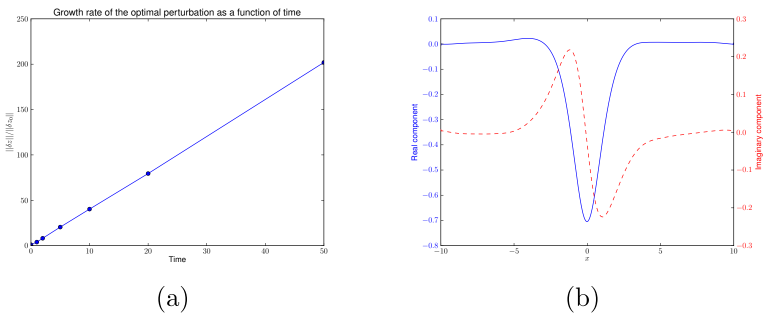

Fig. 4.9. *(a): The growth rate of the optimal perturbation to Gross-Pitaevskii system as a function of time. The optimal perturbations associated with times $T > 10$ are identical. The linear growth of this perturbation was verified using the original nonlinear model up to $T = 500$. (b): The optimal perturbation associated with times $T > 10$. The solid blue line is the real component, while the dashed red line is the imaginary component.*

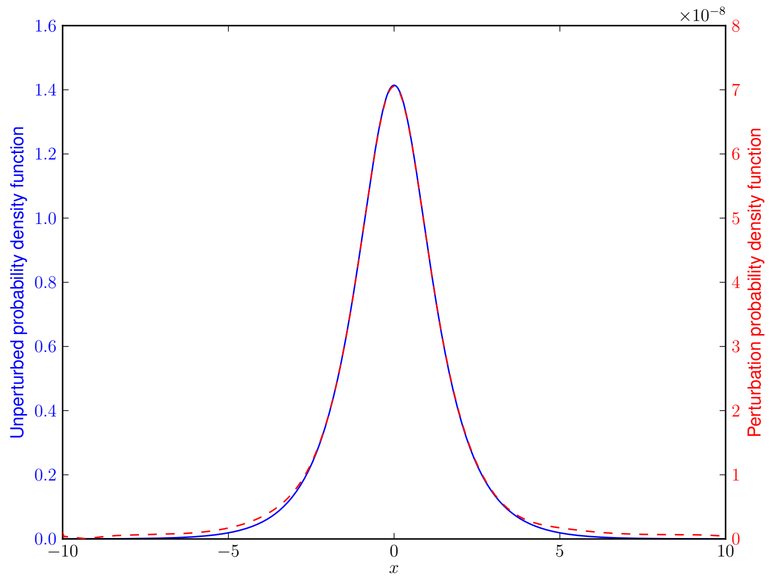

Fig. 4.10. *The probability density functional for the unperturbed Gross-Pitaevskii soliton initial condition (solid blue line) and the optimal perturbation associated with times $T > 10$ (red dashed line). The perturbation corresponds to shifting to a higher (or lower, with negative coefficient) amplitude soliton solution; this is evident since the perturbation has almost the same shape as the soliton itself, but with slightly wider support. Higher (lower) amplitude soliton solutions have greater (lesser) speeds, and so the growth rate is linear in time.*

tangent linear models has major applications in optimisation constrained by partial differential equations, automated error analysis and goal-based adaptivity, continuation and bifurcation analysis, data assimilation, and uncertainty quantification.

A further setting where adjoints prove very useful is Markov Chain Monte Carlo (MCMC) algorithms that are used for Bayesian inference problems. It has been shown that if the derivative of the observation model is available, then the convergence of the algorithm is considerably faster [51, 40]. The derivative is also useful for avoiding getting stuck in local maxima [6]. Bayesian inverse problems have been recently rigorously formulated on function spaces in a well-posed manner; this means that MCMC algorithms can be appropriately modified so that the number

Mesh elements	$N = 480$		$N = 6,000$		$N = 12,000$	
	Runtime (s)	Ratio	Runtime (s)	Ratio	Runtime (s)	Ratio
Forward model	11.84		58.06		109.67	
Tangent linear model (averged)	23.88	3.02	47.13	1.81	55.44	1.51
Adjoint model (averaged)	24.50	3.07	51.63	1.89	58.88	1.54

TABLE 4.6
Timings for the Gross-Pitaevskii simulation for computing the perturbation that grows optimally to $T = 10$. The perturbation is obtained after 16 tangent linear and adjoint model solves. The table shows the run time for the forward and the averaged timings for the tangent linear and adjoint solves. The Newton solver converges on average after two Newton iterations, which means that the optimal ratio is approximately 1.5. With low resolution ($N = 480$), the cost of the linear solves does not dominate the symbolic manipulation; as the mesh is refined ($N = 12,000$), the linear solves become the dominant cost, and the efficiency ratio approaches the optimal value.

of iterations required to converge is independent of mesh resolution [10]. The possibility of automated adjoint generation opens up the possibility of applying these algorithms in a very broad range of applications where they would not otherwise reach.

Another area of particular relevance to this work is the application of techniques from optimal control to transient growth and bypass transition: whereas generalised stability theory accounts for nonnormal effects, such analyses account for both nonnormal and nonlinear effects [42, 27, 28]. These techniques rely fundamentally on the solution of the associated adjoint system to provide the gradient information necessary for the nonlinear optimisation. Future work will be to explore these applications, and extend these techniques to physical systems where their implementation was previously impractical.

Acknowledgements. PEF's work was supported by EPSRC grant EP/I00405X/1 and a Center of Excellence grant from the Norwegian Research Council to the Center for Biomedical Computing at Simula Research Laboratory. SWF's work was supported by EP/I00405X/1, the Grantham Institute for Climate Change and Fujitsu Laboratories of Europe Ltd. The secondary instability in salt fingers was originally discovered by F. P. MacTavish. The authors would like to thank G. N. Wells for providing the Cahn-Hilliard solver.

References.

[1] M. S. ALNÆS, *UFL: a finite element form language*, in Automated Solution of Differential Equations by the Finite Element Method, A. Logg, K. A. Mardal, and G. N. Wells, eds., Springer, 2011, ch. 17, pp. 299–334.

[2] M. S. ALNÆS, A. LOGG, K. B. ØLGAARD, M. E. ROGNES, AND G. N. WELLS, *Unified Form Language: A domain-specific language for weak formulations of partial differential equations*, 2012. arXiv:1211.4047 [cs.MS].

[3] S. BALAY, J. BROWN, K. BUSCHELMAN, V. EIJKHOUT, W. D. GROPP, D. KAUSHIK, M. G. KNEPLEY, L. C. MCINNES, B. F. SMITH, AND H. ZHANG, *PETSc users manual*, Tech. Report ANL-95/11, Argonne National Laboratory, 2011. Revision 3.2.

[4] S. BALAY, W. D. GROPP, L. C. MCINNES, AND B. F. SMITH, *Efficient management of parallelism in object oriented numerical software libraries*, in Modern Software Tools in Scientific Computing, E. Arge, A. M. Bruaset, and H. P. Langtangen, eds., Birkhäuser Press, 1997, pp. 163–202.

[5] D. BARKLEY, H. M. BLACKBURN, AND S. J. SHERWIN, *Direct optimal growth analysis for timesteppers*, International Journal for Numerical Methods in Fluids, 57 (2008), pp. 1435–1458.

[6] A. Beskos, F. J. Pinski, J. M. Sanz-Serna, and A. M. Stuart, *Hybrid Monte Carlo on Hilbert spaces*, Stochastic Processes and their Approximations, 121 (2011), pp. 2201–2230.

[7] S. N. Bose, *Plancks Gesetz und Lichtquantenhypothese*, Zeitschrift für Physik, 26 (1924), pp. 178–181.

[8] J. W. Cahn and J. E. Hilliard, *Free energy of a nonuniform system. I. Interfacial free energy*, The Journal of Chemical Physics, 28 (1958), pp. 258–267.

[9] I. Capuzzo Dolcetta, S. Finzi Vita, and R. March, *Area-preserving curve-shortening flows: from phase separation to image processing*, Interfaces and Free Boundaries, 4 (2002), pp. 325–343.

[10] S. L. Cotter, M. Dashti, and A. M. Stuart, *Approximation of Bayesian inverse problems for PDEs*, SIAM Journal on Numerical Analysis, 48 (2010), pp. 322–345.

[11] J. Crank and P. Nicolson, *A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type*, Mathematical Proceedings of the Cambridge Philosophical Society, 43 (1947), pp. 50–67.

[12] A. Einstein, *Quantentheorie des einatomigen idealen Gases*, Sitzungsberischte der Preussische Akademie der Wissenschaften, (1924).

[13] I. Eisenman, *Non-normal effects on salt finger growth*, Journal of Physical Oceanography, 35 (2005), pp. 616–627.

[14] B. F. Farrell, *The initial growth of disturbances in a baroclinic flow*, Journal of Atmospheric Sciences, 39 (1982), pp. 1663–1686.

[15] ——, *Transient growth of damped baroclinic waves*, Journal of Atmospheric Sciences, 42 (1985), pp. 2718–2727.

[16] B. F. Farrell and P. J. Ioannou, *Generalized stability theory. Part I: Autonomous operators*, Journal of the Atmospheric Sciences, 53 (1996), pp. 2025–2040.

[17] ——, *Generalized stability theory. Part II: Nonautonomous operators*, Journal of Atmospheric Sciences, 53 (1996), pp. 2041–2053.

[18] C. P. Farrell, *Simulating ultracold matter: horizons and slow light*, PhD thesis, University of St. Andrews, 2008.

[19] P. E. Farrell, S. W. Funke, D. A. Ham, and M. E. Rognes, *Automated derivation of the adjoint of high-level transient finite element programs.* Submitted to SIAM Journal on Scientific Computing, 2012. arXiv:1204.5577 [cs.MS].

[20] M. Förster, U. Naumann, and J. Utke, *Toward adjoint OpenMP*, tech. report, RWTH Aachen, 2011. AIB-2011-13.

[21] P. Gray and S. K. Scott, *Autocatalytic reactions in the isothermal, continuous stirred tank reactor: isolas and other forms of multistability*, Chemical Engineering Science, 38 (1983), pp. 29–43.

[22] ——, *Autocatalytic reactions in the isothermal, continuous stirred tank reactor: oscillations and instabilities in the system $A + 2B \to 3B$; $B \to C$*, Chemical Engineering Science, 39 (1984), pp. 1087–1097.

[23] A. Griewank, *Evaluating derivatives: principles and techniques of algorithmic differentiation*, Frontiers in Applied Mathematics, SIAM, 2008.

[24] E. Gross, *Structure of a quantized vortex in boson systems*, Il Nuovo Cimento, 20 (1961), pp. 454–477.

[25] V. Hernández, J. E. Román, A. Tomás, and V. Vidal, *Krylov-Schur methods in SLEPc*, Tech. Report STR-7, Universitat Politècnica de València, 2007.

[26] V. Hernandez, J. E. Roman, and V. Vidal, *SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems*, ACM Transactions on Mathematical Software, 31 (2005), pp. 351–362.

[27] M. P. Juniper, *Transient growth and triggering in the horizontal Rijke tube*, International Journal of Spray and Combustion Dynamics, 3 (2011), pp. 209–224.

[28] ———, *Triggering in the horizontal Rijke tube: non-normality, transient growth and bypass transition*, Journal of Fluid Mechanics, 667 (2011), pp. 272–308.

[29] E. Kalnay, *Atmospheric Modeling, Data Assimilation and Predictability*, Cambridge University Press, 2002.

[30] R. C. Kirby and A. Logg, *A compiler for variational forms*, ACM Transactions on Mathematical. Software, 32 (2006), pp. 417–444.

[31] O. Lahav, A. Itah, A. Blumkin, C. Gordon, S. Rinott, A. Zayats, and J. Steinhauer, *Realization of a sonic black hole analog in a Bose-Einstein condensate*, Physical Review Letters, 105 (2010), p. 240401.

[32] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, Journal of Research of the National Bureau of Standards, 45 (1950), pp. 255–282.

[33] R. Leine, *The historical development of classical stability concepts: Lagrange, Poisson and Lyapunov stability*, Nonlinear Dynamics, 59 (2010), pp. 173–182.

[34] A. Logg, K. A. Mardal, G. N. Wells, et al., *Automated solution of differential equations by the finite element method*, Springer, 2011.

[35] A. Logg and G. N. Wells, *DOLFIN: Automated finite element computing*, ACM Transactions on Mathematical Software, 37 (2010).

[36] E. N. Lorenz, *A study of the predictability of a 28-variable atmospheric model*, Tellus, 17 (1965), pp. 321–333.

[37] A. M. Lyapunov, *The General Problem of the Stability of Motion*, Control Theory and Applications Series, Taylor & Francis, 1892. Translated by A. T. Fuller.

[38] X. Mao, S. Sherwin, and H. Blackburn, *Transient growth and bypass transition in stenotic flow with a physiological waveform*, Theoretical and Computational Fluid Dynamics, 25 (2011), pp. 31–42.

[39] G. R. Markall, A. Slemmer, D. A. Ham, P. H. J. Kelly, C. D. Cantwell, and S. J. Sherwin, *Finite element assembly strategies on multi-core and many-core architectures*, International Journal for Numerical Methods in Fluids, (2012).

[40] J. Martin, L. Wilcox, C. Burstedde, and O. Ghattas, *A stochastic Newton MCMC method for large-scale statistical inverse problems with application to seismic inversion*, SIAM Journal on Scientific Computing, 34 (2012), pp. A1460–A1487.

[41] O. K. Matar and S. M. Troian, *The development of transient fingering patterns during the spreading of surfactant coated films*, Physics of Fluids, 11 (1999), pp. 3232–3246.

[42] A. Monokrousos, A. Bottaro, L. Brandt, A. Di Vita, and D. S. Henningson, *Nonequilibrium thermodynamics and the optimal path to turbulence in shear flows*, Physical Review Letters, 106 (2011), p. 134502.

[43] A. M. Moore, H. G. Arango, E. Di Lorenzo, B. D. Cornuelle, A. J. Miller, and D. J. Neilson, *A comprehensive ocean prediction and analysis system based on the tangent linear and adjoint of a regional ocean model*, Ocean Modelling, 7 (2004), pp. 227–258.

[44] U. Naumann, *The Art of Differentiating Computer Programs*, Software, Environments and Tools, SIAM, 2011.

[45] K. B. Ølgaard and G. N. Wells, *Optimizations for quadrature representations of finite element tensors through automated code generation*, ACM Transactions on Mathematical Software, 37 (2010), pp. 8:1–8:23.

[46] T. M. Özgökmen and O. E. Esenkov, *Asymmetric salt fingers induced by a nonlinear equation of state*, Physics of Fluids, 10 (1998), pp. 1882–1890.

[47] T. N. Palmer, *Medium and extended range predictability and stability of the Pacific/North*

American mode, Quarterly Journal of the Royal Meteorological Society, 114 (1988), pp. 691–713.

[48] P. C. Parks, *A. M. Lyapunov's stability theory—100 years on*, IMA Journal of Mathematical Control and Information, 9 (1992), pp. 275–303.

[49] J. E. Pearson, *Complex patterns in a simple system*, Science, 261 (1993), pp. 189–192.

[50] L. P. Pitaevskii, *Vortex lines in an imperfect Bose gas*, Soviet Physics JETP, 13 (1961), pp. 451–454.

[51] G. O. Roberts and R. L. Tweedie, *Exponential convergence of Langevin distributions and their discrete approximations*, Bernoulli, 2 (1996), pp. 341–363.

[52] P. J. Schmid, *Nonmodal stability theory*, Annual Review of Fluid Mechanics, 39 (2007), pp. 129–162.

[53] M. E. Stern, *The "salt-fountain" and thermohaline convection*, Tellus, 12 (1960), pp. 172–175.

[54] G. W. Stewart, *A Krylov–Schur algorithm for large eigenproblems*, SIAM Journal on Matrix Analysis and Applications, 23 (2001), pp. 601–614.

[55] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*, Society for Industrial Mathematics, 1997.

[56] L. N. Trefethen and M. Embree, *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*, Princeton University Press, 2005.

[57] L. N. Trefethen, A. E. Trefethen, S. C. Reddy, and T. A. Driscoll, *Hydrodynamic stability without eigenvalues*, Science, 261 (1993), pp. 578–584.

[58] S. Tremaine, *On the origin of irregular structure in Saturn's rings*, The Astronomical Journal, 125 (2003), p. 894.

[59] J. S. Turner, *Multicomponent convection*, Annual Review of Fluid Mechanics, 17 (1985), pp. 11–44.

[60] W. G. Unruh, *Experimental black-hole evaporation?*, Physical Review Letters, 46 (1981), pp. 1351–1353.

[61] J. Utke, L. Hascoet, P. Heimbach, C. Hill, P. Hovland, and U. Naumann, *Toward adjoinable MPI*, in Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on, Rome, Italy, 2009, pp. 1–8.

[62] G. N. Wells, E. Kuhl, and K. Garikipati, *A discontinuous Galerkin method for the Cahn–Hilliard equation*, Journal of Computational Physics, 218 (2006), pp. 860–877.

[63] K. Wu and H. Simon, *Thick-restart Lanczos method for large symmetric eigenvalue problems*, SIAM Journal on Matrix Analysis and Applications, 22 (2000), pp. 602–616.

[64] L. Zanna, P. Heimbach, A. M. Moore, and E. Tziperman, *Optimal excitation of interannual atlantic meridional overturning circulation variability*, Journal of Climate, 24 (2011), pp. 413–427.

[65] ——, *Upper-ocean singular vectors of the North Atlantic climate with implications for linear predictability and variability*, Quarterly Journal of the Royal Meteorological Society, 138 (2012), pp. 500–513.